SLIM Howto (beta)

Bruno Bzeznik

May 21, 2008

Contents

Chapter 1

Introduction

SLIM is the "SLIS Management" application. It's goal is to provide a database containing all the informations necessary to deploy and maintain a park of SLIS servers, and so much. In the database, you'll not only put your SLIS servers, but all of your ip addresses, networks, DNS entries, hardware configurations, MAC addresses, ... all of the things that describes your WAN/MAN, but not the LANs that are behind de SLIS servers.

The relational database is built over PostgreSQL and the interface to this database is built over PHP. So you can access your database from a browser using https. You can create users that will have limited access to some data and functions. SLIM will automaticaly generate the setup disks (DKS) necessary for the SLIS installations. The users entering the data and the user generating a DKS may be different people using SLIM at different time, and informating each other by the way of "statuses" of a SLIS entry. SLIM also allows you to generate configs for some of your central servers (DNS, VPN, Mail concentrator). A complex "default values system" allows you to customize the generated configuration files and help you to speed up the process of entering data.

SLIM also aims to provide a way to manage incidents tickets, so it can be a part of an assistance call center.

Monitoring is not a part of the "SLIM application", but it is a part of the "SLIM project", in the form of a plugin that will have a specific section into this document.

Chapter 2

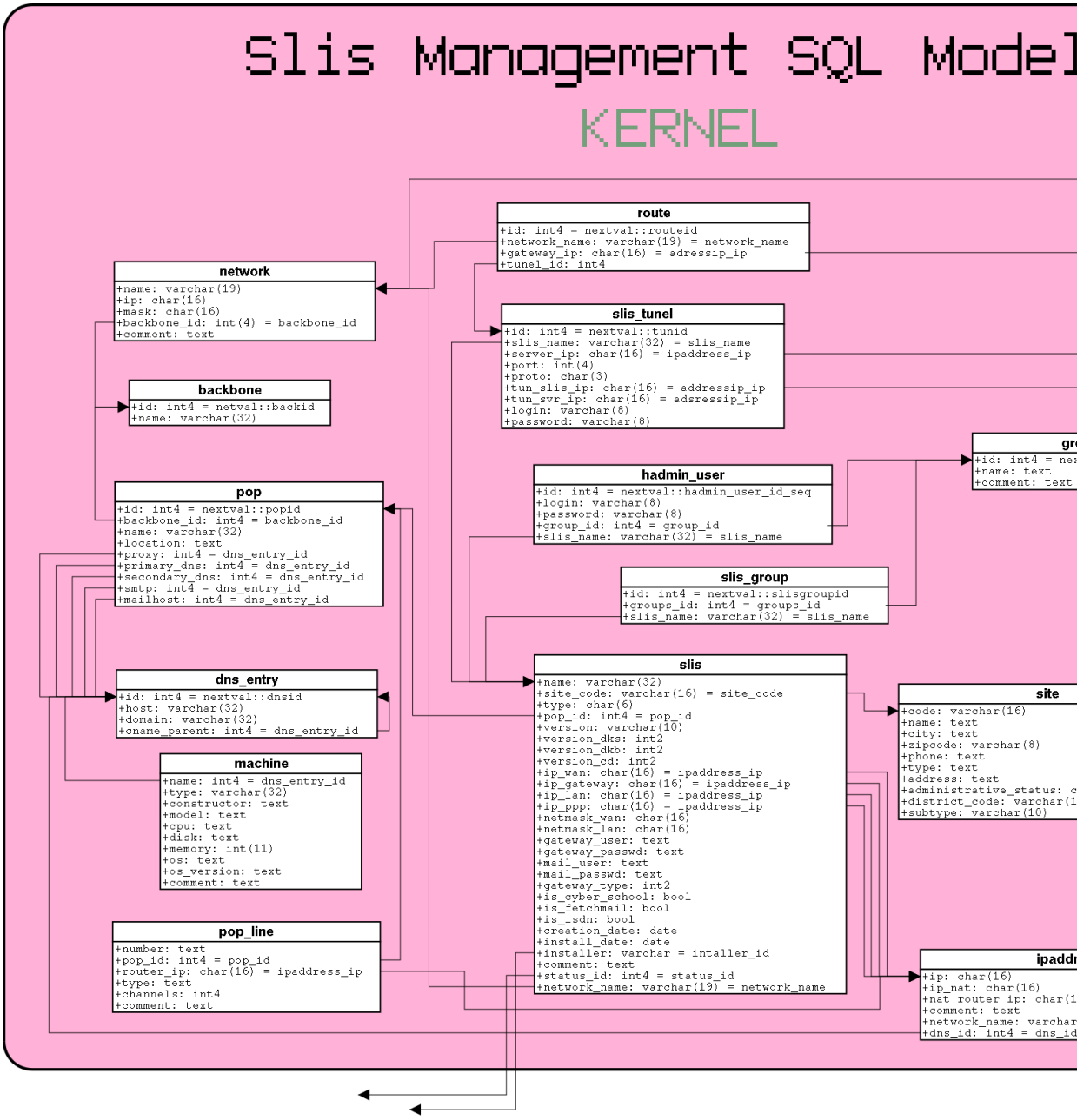
Application architecture

If you are the SLIM administrator, you MUST understand the architecture of this application, especially the SQL model that is its kernel. Because you'll probably have to use advanced functions to customize SLIM for your domain, you'll have to refer to the SQL model, often to know the name of the fields and tables. You'll probably also have to know a bit about the SQL syntax.

2.1 SQL model

2.1.1 The kernel

What we call 'the kernel of the SQL model' is composed of all the tables containing the essential data. It contains the ip addresses, the network description, the sites, the dns entries, and of course, the SLIS data.



The kernel of the SQL model. This diagramm may be found in different format at the developpment repository
https://svn.slis.fr/svn/slis/slim/sources/trunk/slis-slim/docs/howto/

FIELDS DESCRIPTION

- **Table: slis**

  This table contains the entities describing SLIS servers

  - **name (primary key)** The name of the SLIS. We recommend to use the DNS name of the slis host, without the domain. It must be unique for all the SLIS contained into the SLIM database.
  - **site_code** The code of the site to witch the SLIS is attached. A SLIS is always attached to a *site*. There might be several SLIS for one given site. See the *site* table.
  - **type** This allows you to categorize the SLIS. The type is used by the default system as a key to define the default values for pre-entered fields, variables of config files or dns entries.
  - **pop_id** The key of the *pop* the SLIS belongs to. See the *pop* table.
  - **version** The version number of the SLIS (ie: 3.0)
  - **version_dks** The sub-version number of the setup disk (ie: 3). The version and version_dks values are used by the DKS center to construct the name of the directory where to find the DKS matrix (ie v3.0dks3)
  - **version_dkb** The version number of the boot disk. Only informational, not used.
  - **version_cd** The version number of the CD. Only informational, not used.
  - **ip_wan** The ip address of the SLIS on its ethernet interface to the WAN side. It's often unique, but it is not mandatory: for a PPPOE SLIS, it may be 10.0.0.10.
  - **ip_gateway** The ip address of the default gateway to the internet. It may be empty as the gateway of a PPPOE or DHCP SLIS is dynamically given.
  - **ip_lan** The ip address of the SLIS on its ethernet interface to the LAN side. It may be the same for every SLIS (ie: 172.16.0.1)
  - **ip_ppp** In the case of a PPPOE SLIS, this is the ip address on the WAN side, if this ip is fixed. In spite this ip might be affected dynamically, if it is fixed, this is often usefull to keep it in the database. Then, it may be used by the default system to configure the DNS.
  - **netmask_wan** The netmask on the wan interface.
  - **netmask_lan** The netmask on the lan interface.
  - **gateway_user** The login of the router or the pppoe account. This field may be empty if the router is not managed by the SLIS.
  - **gateway_passwd** The password associated to gateway_user.
  - **mail_user** The user of the POP account in the case of the FETCHMAIL option. See the comments into the `setup.data` file.
  - **mail_passwd** The password associated to mail_user.

- **gateway_type** A number corresponding to the router used for the gateway to Internet. For a non supported router or a permanent link on wich the router's type is unknown, this number is '0'. The supported routers and corresponding numbers are listed into the comments about the GATEWAY_TYPE variable into the `setup.data` file.
- **is_cyber_school** A boolean representing if the 'CyberSchool' option is active or not.
- **is_fetchmail** A boolean for the 'fetchmail' option.
- **is_isdn** A boolean for the 'isdn' option.
- **creation_date** A date telling when the SLIS entry is born.
- **install_date** A date telling when the SLIS has been first installed.
- **installer** A relation to the id of the 'installer'
- **comment** A free text field.
- **status_id** The id of the status in which the SLIS is.
- **network_name** The name of the network on wich the SLIS is attached by its WAN side. There may be several SLIS on the same network. It's up to you to be carefull about the ip_lan: each SLIS must have one on the specified network, and it must be unique on the given network since this network is a subnet of a bigger network.

- **Table: site**

  This table contains the sites definitions. A SLIS always depends on a site.

  - **code (primary key)** A unique code identifying the site.
  - **name** The name of the site.
  - **city** The city in which the site resides.
  - **zipcode** The postal code of the city of the site.
  - **phone** The phone number of the site.
  - **type** The type used to classify the site. It may be 'school', 'hischool', 'college', etc. for example.
  - **address** The address (number, street) of the site.
  - **administrative_status** Another field used to classify the site. This one is intented to tell if the site is 'private' or 'public' but, as it is informational only, you can use it for whatever else you want.
  - **district_code** This is a relation to the code of another site. By this way, you can create a hierarchy of sites. The linked site is called a 'district' as it may be an entity grouping several sites.
  - **subtype** Another field used to classify the site. It is intented to be a type into the type.

- **Table: ip_address**

  This table contains ip addresses

  - **ip (primary key)** The ip address in decimal notation. Example: 10.0.0.1
  - **ip_nat** If this address is staticaly NATed, this is a relation to the NATed ip. Example: 193.54.149.1
  - **nat_router_ip** Informational field only. It gives the ip of the router making the translation.
  - **comment** A free text field.
  - **network_name** The name of the network (or subnet) the ip belongs to.
  - **dns_id** A relation to the id of a dns_entry. This field associates names to ips and is used for the DNS server configuration.

- **Table: network**

  This table contains network definitions

  - **name (primary key)** This is the name of the network, in CIDR notation. Example: 10.0.0.0/24
  - **ip** The ip of the network. Example: 10.0.0.0
  - **mask** The mask in decimal notation. Exemple: 255.255.255.0
  - **backbone_id** The relation to the id of the backbone the network belongs to.
  - **comment** A free text field.

- **Table: backbone**

  This table contains backbones definitions

  - **id (primary key)** A unique number identifying a backbone.
  - **name** The name of the backbone. A backbone, into SLIM, is not more than a group of pops.

- **Table: pop**

  This table contains the 'points of presence' definitions

  - **id (primary key)** A unique number identifying a pop.
  - **backbone_id** The id of the backbone to which the pop is attached. This is for grouping pops.
  - **name** The name given to this pop.
  - **location** The place where the pop leave. This is only an information, without any relation.
  - **proxy** The DNS name (host.domain) of the proxy server if there is one (may be empty). This information is used by the DKS center, when generating the setup disk (variable PROXY_CACHE of the setup.data file). This value may be erased by the defaults set up into the file defaults.

- **primary_dns** The DNS name (host.domain) of the primary DNS server. This information is used by the DKS center, when generating the setup disk (variable DNS_PRIMARY of the setup.data file). There must be one ip address associated to this name, because this is this ip that will be used into the DKS, and not the name. This value may be erased by the defaults set up into the file defaults.
- **secondary_dns** Same thing of primary_dns (but for the DNS_SECONDARY setup.data variable)
- **smtp** The DNS name (host.domain) of the smtp server that a SLIS connected to this pop will use to relay the mail. This information is used by the DKS center, when generating the setup disk (variables SMTP_OUT and SMTP_OUT_IP of the setup.data file). There must be one ip address associated to this name, because the name AND the ip are declared. This value may be erased by the defaults set up into the file defaults.
- **mailhost** The DNS name (host.domain) of the smtp server that will send the mail to a SLIS connected to this pop. This information is used by the DKS center, when generating the setup disk (variables SMARTHOST and SMARTHOST_IP of the setup.data file). There must be one ip address associated to this name, because the name AND the ip are declared. This value may be erased by the defaults set up into the file defaults.

- **Table: dns_entry**

  This table contains the DNS names that may be associated to ip addresses

  - **id (primary key)** A unique number identifiying the DNS entry.
  - **host** The host part of the DNS entry.
  - **domain** The domain part of the DNS entry.
  - **cname_parent** A relation to the id of another DNS entry. This means that the current entry is an alias to this linked entry.

- **Table: pop_line**

  This table contains the numbers of ISDN lines or similar things associated to a pop

  - **number (primary key)** The number of the line (example: 0476150003)
  - **pop_id** A relation to the id of the pop where the line leaves. The line number and pop will be used by the DKS center when generating router configuration files. A SLIS connected to this pop will have its router configured to used the associated line number. If several lines are associated to a given pop, only the first number is used (in the order of creation).
  - **router_ip** The ip address of the concentrator this line is attached to. This is only informational.
  - **type** A free string representing the type (example: T2)

  - **channels** The number of channels.
  - **comment** A free text field.

- **Table: machine**

  This table contains hardware informations about hosts

  - **name (primary key)** The DNS name of a host. This is a unique key linked to the dns_entry table. So, we attach a physical host to a DNS name.
  - **type** A string giving the type of host (examples: "router", "server").
  - **constructor** The name of the constructor (examples: "Cisco", "IBM", "HP")
  - **model** The model of the host (examples: "3661", "e230")
  - **cpu** The CPU speed (example: "2Ghz")
  - **disk** The disk capacity (example: "36GB")
  - **memory** The memory size in megabytes (example: "512")
  - **os** The name of the operating system installed on this host (example: "Linux RedHat")
  - **os_version** The version number of the operating system (example: "7.3")
  - **comment** A free text field.

- **Table: route**

  This table contains networks to be routed to a tunel (slis_tunel) or a gateway (ip)

  - **id (primary key)** A unique number representing the route.
  - **network_name** A link to the network entry specifying the route.
  - **gateway_ip** The gateway where the above network is routed. May be empty: then, a tunel_id must be specified.
  - **tunel_id** The id of the tunel to which the network is routed.

- **Table: slis_tunel**

  This table contains PTP tunels parameters for a given SLIS

  - **id (primary key)** A unique number identifying the tunel.
  - **slis_name** A link to the name of the SLIS this tunel is attached to.
  - **server_ip** The ip address of the VPN concentrator.
  - **type** A string which is one of "vtun" or "ipsec". This is to know which option to use in the setup.data file (variable VTUN or IPSEC).
  - **port** In the case of a vtun tunel, the server listen on a particular port (5000 by default). Put here the port number used.
  - **proto** In the case of a vtun tunel, the tunel may be built over TCP or over UDP. This field is one of the strings "tcp" or "udp".

- **tun_slis_ip:** The ip of the tunel interface of the SLIS.
- **tun_svr_ip** The ip of the tunel interface of the server, for the given tunel.
- **login** In the case of a vtun tunel, this is a string representing the "login".
- **password** This is a string containing, the password of the vtun tunnel, or the shared key of the ipsec tunel.

- **Table: hadmin_user**

  This table contains initial logins and passwords of the SLIS interface

  - **id (primary key)** A unique number identifying the user.
  - **login** A string containing the login. This login will be useable to log on the SLIS administration interface when asked for http auth. (the one which is behind the 1098 tcp port)
  - **password** A string containing the password associated to the login.
  - **group_id** This user may be replicated on every SLIS of a given group. To do so, you put the id of the SLIS group here. This is a relation to the "group" table. When this filed is used, there's no need to fill the "slis_name" field.
  - **slis_name** The name of the SLIS on which the hadmin user resides.

- **Table: slis_group**

  This table contains groups of SLIS

  - **id (primary key)** A unique number representing the group affiliation.
  - **groups_id** A relation to the id of a group.
  - **slis_name** The name of a slis that belongs to the group which id is "groups_id".

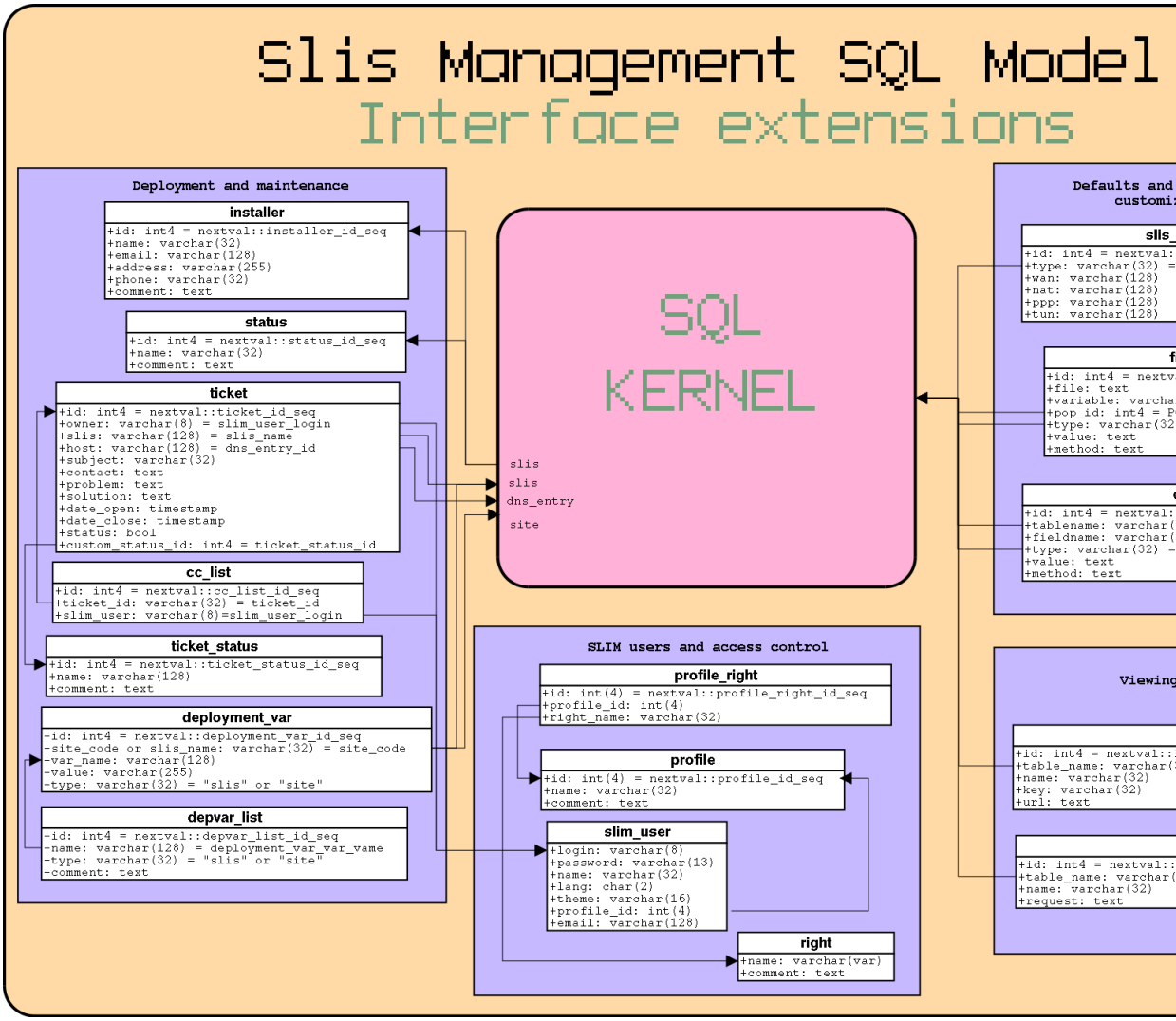
- **Table: group**

  This table contains SLIS groups definitions

  - **id (primary key)** A unique number identifying the group.
  - **name** A string that is the name of the group.
  - **comment** A free text field.

2.1.2 The interface extensions

The 'interface extensions of the SQL model' contains tables that are used for the interface behaviors or the help to the support. The relations with the kernel may be on table names or main indexes such as the slis names or the dns entries. This part of the model contains data related to the users and access controls of SLIM, the support, the viewing capabilities and the default values' system.



The extensions of the SQL model . This diagramm may be found in different format at the developpment repository
https://svn.slis.fr/svn/slis/slim/sources/trunk/slis-slim/docs/howto/

FIELDS DESCRIPTION

- **Table: slim_user**

  This table contains the users that will be able to connect to SLIM

  - **login**
  - **password**
  - **name**

  - **lang**
  - **theme**
  - **profile_id**
  - **email**

- **Table: profile**

  This table contains users profiles definitions

  - **id**
  - **name**
  - **comment**
  - **profile_type** The type of profile (0: global , 1: group)

- **Table: profile_right**

  This table contains the rights that a profile gives to users

  - **id**
  - **profile_id**
  - **right_id**

- **Table: right**

  This table contains rights defintions

  - **id**
  - **name**
  - **comment**
  - **type** The type of profile (0: global , 1: group)

  Table: child_rights

  This table contains a single level of right heritage

  - **id**
  - **p_right_id** The id of the parent right
  - **c_right_id** The id of the child right

  Table: slim_rights

  This table contains the 'groups profile' for a user on a 'slis group'

  - **id**
  - **slim_user_login** The login of the Slim user
  - **group_id** The group on which the user has rights contained in the 'profile_id' profile

  - **profile_id** The group profile that contains the rights apply on the slis group 'groups_id' for the user 'slim_user_login'

- **Table: slis_dns_default**

  This table contains the DNS defaults matrix

  - **id**
  - **type**
  - **wan**
  - **nat**
  - **ppp**
  - **tun**

- **Table: defvalue**

  This table contains the default values for the fields of some forms

  - **id**
  - **table_name**
  - **field_name**
  - **typet**
  - **value**
  - **method**

- **Table: file_defvalue**

  This table contains the default values for variables into a config file

  - **id**
  - **file**
  - **variable**
  - **pop_id**
  - **type**
  - **value**
  - **method**

- **Table: installer**

  This table contains the SLIS installers definitions

  - **id**
  - **name**
  - **email**
  - **address**

  - **phone**
  - **comment**

- **Table: status**

  This table contains the statuses of SLIS definitions

  - **id**
  - **name**
  - **comment**

- **Table: ticket**

  This table contains the technical hitches tickets, used for the support

  - **id**
  - **owner**
  - **slis**
  - **host**
  - **subject**
  - **contact**
  - **problem**
  - **solution**
  - **date_open**
  - **date_close**
  - **status**
  - **custom_status_id**

- **Table: cc_list**

  This table contains the users associated to a ticket

  - **id**
  - **ticket_id**
  - **slim_user**

- **Table: ticket_status**

  This table contains the definitions of custom statuses of tickets

  - **id**
  - **name**
  - **comment**

- **Table: deployment_var**

  This table contains the 'help to deployment' variables

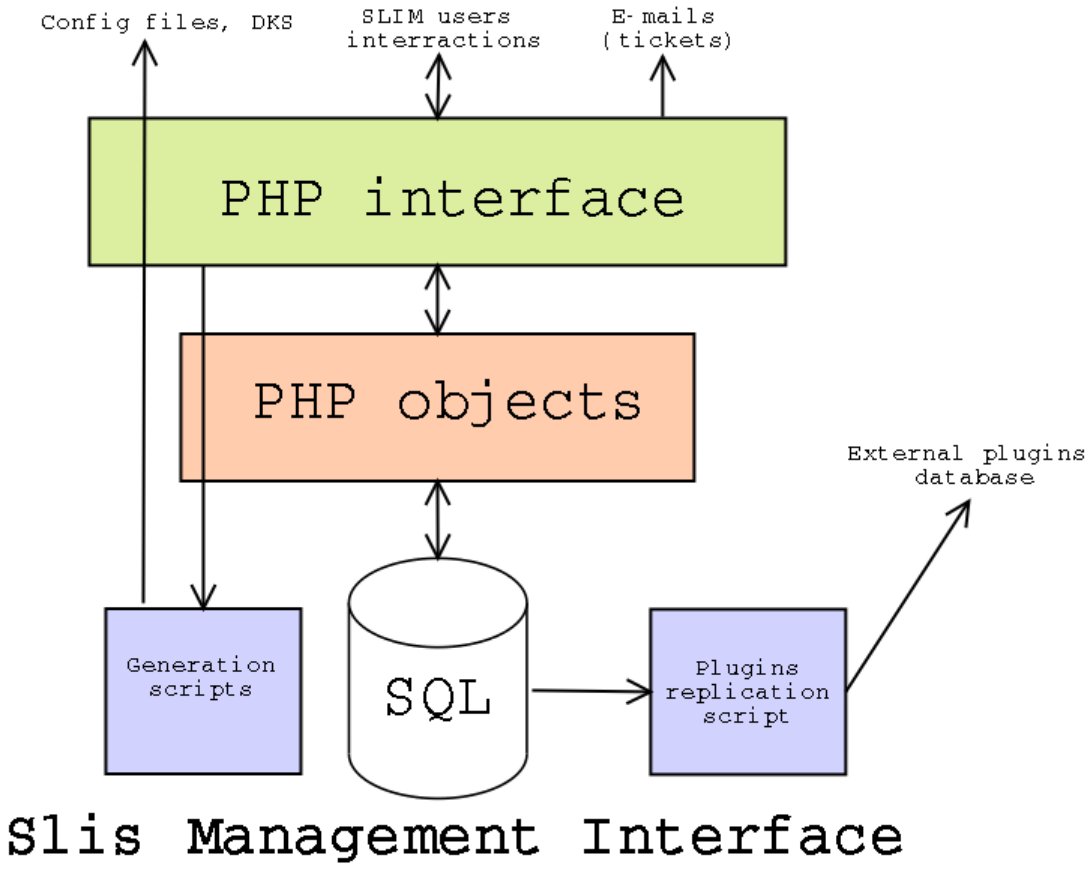
  - **id**
  - **site_code**
  - **var_name**
  - **value**
  - **type**

- **Table: depvar_list**

  This table describes the possible values for deployment_vars

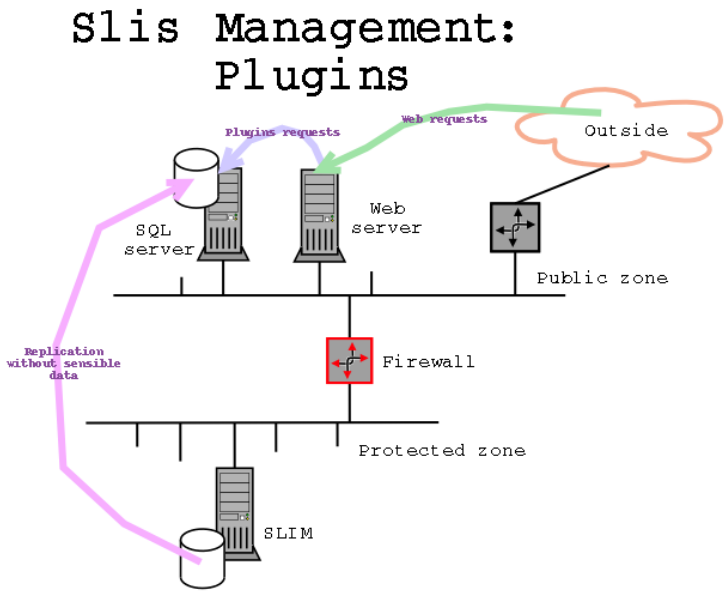
  - **id**
  - **name**
  - **type**
  - **comment**

2.2 PHP interface



The SLIM interface. This diagramm may be found in different format at the developpment repository https://svn.slis.fr/svn/slis/slim/sources/trunk/slis-slim/docs/howto/

2.3 Plugins



The SLIM plugins secure implementation . This diagramm may be found in different format at the developpment repository
https://svn.slis.fr/svn/slis/slim/sources/trunk/slis-slim/docs/howto/

For plugins introduction and the way to install the plugins system, see the section called Plugins into the Installation chapter.

2.3.1 The monitoring plugin

To do...

Chapter 3

Installation and configuration

3.1 Prerequisite

SLIM is now packaged for the Debian GNU/Linux operating system. You must have a running Debian Etch - GNU/Linux system to install the slis-slim package. It's preferable to use this operating system only for the SLIM.

Intalling the package slis-slim will automatically install the following packages:

```
- apache2-mpm-prefork
- postgresql-8.1
- postgresql-contrib-8.1
- libapache2-mod-auth-pgsql
- php5
- php5-pgsql
- php5-curl
- php5-gd
- curl
- cracklib-runtime
- libnetaddr-pg-perl
- libnetaddr-ip-perl
- apg
- wget
- zip
- wfrench
- wamerican
```

3.1.1 Backuping before upgrading

It's preferable to backup all the slim datas before starting to upgrade it. The datas are both on the postgresql databases ('slim' and 'slismonitor') and on the filesystem.

- **The databases :** The easiest way to backup the slim databases is to dump the two entire databases. You can use the pg_dump command like this :

```
su postgres -c "pg_dump -Ft -b <database> > <filename>"
```

To restore the database, you can the pg_restore command like this :

```
su postgres -c "pg_restore -d <database> <filename>"
```

- **The filesystem :** Don't forget to backup *the SSL certificates*, *the user scripts* (If the slim was modified), *the user system scripts* (like the dns configuration script) and *the slim configuration*.

3.2 In case of upgrading

Be carefull, *you have to backup the databases and the parts of the filesystem corresponding to the slim before upgrading.*

Before the version 2.0.5 , upgrading slim with the debian packaging system isn't possible. You must backup the slim datas, install a 'Debian Etch' operating system, install the new slis-slim package and restore manually the datas.

After the version 2.0.5, the slis-slim upgrade can be done by the debian package system. You just have to read the version notes and upgrade with 'dpkg' or 'apt-get'.

3.3 In case of a new installation

The slis-slim package can be installed with the debian package system. You just need to use 'dpkg' and follow the questions to fit your needs.

3.3.1 The SSL certificates

Because SLIM gather sensible datas, it's recommanded to use a secure web server. The SSL mode for apache web server need a SSL certificate to encrypt the datas. When the package is installed, you will be asked if you want to use an existing SSL certificate. It's preferable to have a certificate signed by a recognized authority but if you haven't you can create one self-signed.

If you have your own certificate, after the package installation, you have to put it on the right place (for exemple here : `/etc/slis-slim/ssl`) and modify the configuration file `/etc/slis-slim/config.php` and the `/etc/slis-slis/apache.conf` to use it. Don't forget to restart the apache2 service.

For more security, you can also separate the SSL certificate used by the web server and the one used by the slis monitoring layout. In this case you have to specify the right *private* key in the `/etc/slis-slim/config_monitor.php` and create the symlink `/var/www/slim/scripts/monitor/get/slimcert.pem` to the right *public* certificate.

3.3.2 Apache and php configuration

The installation process will automatically configure apache2 and php5. Nevertheless two operations need to be validated by the operator. The first is the change of the apache listening port to 443. The second is the change of the php configuration to fit the slim needs. If these are not validated, you will have to setup yourself a right configuration to make slim work.

3.3.3 The postgres 'slim' and 'slismonitor' databases

The main datas are stored in the two databases 'slim' an 'slismonitoring'.

3.3.3.1 The 'slim' database

This database is necessary to make slim work. It gather all the information that are persistent. The installation process can setup postgresql, create the user and the databse 'slim' and fill it with data samples. All these steps will be asked during the installation process.

You will find the information about host, database name, user, pass for this database in the file `/etc/slis-slim/config.php`. You will also find the SQL scripts used for creating the database structures and for adding the data samples in `/usr/share/slis-slim/`.

3.3.3.2 The 'slismonitor' database

This database gather all the volatile information about the slis monitoring. For security reason and for performance, it is better to not put it on the same host. In fact this database tend to grow as fast as the number of monitoring scripts, the number of slis and the number of values stored grow.

During the installation process, you will be asked if you want to use a database hosted by another operating system. In this case you will need to give the host address, database name, user and password to use. And then choose if you want to create the database structure and fill it with the data samples.

In case of using the local host for the 'slismonitoring' database. The users 'slimmonitoring' and 'slismonitoring' will be automatically created and, according to your choice during the installation process, the database will be created and filled with the data samples.

3.3.4 Installing a mail transport agent

The operating system must be prepared to be able to send email : you need to install an MTA like postfix, exim, etc..

3.4 The configuration files

The configuration files has been gathered into one directory `/etc/slis-slim/`.

- `apache.conf` : gather the configuration for Apache 2.2
- `php_slim.ini` : gather the configuration for PHP 5. SLIM is designed to work with some specific php parameters.
- `htaccess` : gater the configuration for the slim web interface access.
- `config.php` : gather the main configuration options for the SLIM.
- `config_monitor.php` : gather the main configuration options for the SLIM monitoring.
- `ssl` : contain the SSL keys and certificates for SLIM.

3.5 The system scripts

The SLIM architecture give the possibility to extract easily information with simple SQL requests. It could be usefull to use the slim datas for updating automatically the configuration of the nameserver, the vpn server, the radius server, etc.. To do that there are some scripts as examples in `/usr/share/doc/slis-slim/`.

You can easily use them and modify them to fit your needs.

3.6 Plugins

3.6.1 Introduction to SLIM plugins

You may want to use some data from SLIM with other applications or just to print out public informations, like the number of installed SLIS, on a public web server. But you need to do that in a secure way, and want to be sure that stored passwords into your SLIM server won't be stollen.

The way to do that, is to make those applications use another database, on another server preferably. This database will be a copy of the real SLIM database, but without sensitive informations. We will call the applications accessing this copy the "plugins" and the copied database will be the "plugins-database". The server hosting the plugins-database will be called the "plugins-server".

SLIM comes with a script "replicate-plugins-database.bash" that will recreate the plugins-database each time you start it. You may run it by the crontab on your SLIM host. You have to edit the top of this script, to configure it.

What SLIM does not do:

- It will not create users and users privileges on the plugins-database server. It's up to you to manage those users and configure the plugins with them.
- It will not set up your crontab
- It will not set up the web server and the sql server hosting the plugins and the plugins-database

By this way, your are responsible of the security of the data you store into SLIM.

3.6.2 How to set up the SLIM plugins system

- You must have a running postgresql server on another host (the "plugins-server"). Same thing for a running http server, running php (or whatever the plugins will need to interoperate with you or other things...) It's possible to run the plugins-database on the SLIM host, but not recommended. Ensure that your SLIM host may connect to the postgresql port of the plugins-server.
- Set up your plugins-server:
  - Create a user, for example "slimplugins" with createdb privilege:

```
template1=# create userslimplugins with password
    'MYPASS' createdb nocreateuser;
```

  - Set up the `pg_hba.conf` file to allow your SLIM-host to connect via this user with the MD5 method:

```
host         all         slim.mydomain.com  ↩
    255.255.255.255
      md5
```

    Don't forget to restart postgres.
  - Set up the users for the plugins. For example:

```
template1=# create userslimstatus with password
    'ROUSER' nocreatedbnocreateuser;
```

    The grants will be put into `replicate-plugins-database.bash`.
- Edit the top of the script `replicate-plugins-database.bash` into your SLIM directory.
- Move this script to `/usr/local/sbin` and protect it with `chmod 700`
- Start replicate-plugins-database.bash on the SLIM-host and set up your plugins on the other(s) host(s).

3.6.3 Provided plugins

You'll find some plugins into the SLIM distribution, in the `plugins` directory. You just have to put the one you want to use onto a web server using php (with postgres libs compiled in) and configure it by editing `config.inc.php`. The webserver must be able to contact the plugins server on the postgresql port.

3.7 Security

The first thing you MUST do when you just installed your SLIM host is to put your apache server in SSL mode. By this way, you'll connect with the https protocol that is much more secure than http. But this requires that you manage a certificate for your

server. This is why this is not done by the automatic installation. To learn more about mod_ssl, take a look at http://www.modssl.org/

Also note that if you use an SQL client to connect to the SLIM database directly from the network, data may be transmitted clear over the network. This is why we recommend the use of PhpPgAdmin, encapsulated over https. See the section Feed the database with an SQL client for more informations.

Chapter 4

User's guide

4.1 Introduction to the SLIM interface

First of all, you need to connect to the interface as the 'admin' user. This user has the default password 'slim'.

The interface is divided in two frames. On the left, the menu; on the right, the main frame. The menu contains 3 icones (depending of the theme used) at the top. The first one is to logout and then login with another username. The second is for expanding or unexpenanding the menu. The last is to call the welcome page again. The menu is composed of sections and links into sections. Once you have expanded the menu, you'll see new sections and new links, appearing in italic. Each section or link is preceded by a little icone. If you let the pointer of your mouse one second on this icone, you'll see a short text explaining what the section or the link does. The menu is divided in 4 parts, separated by an horizontal line. The first part contains sections allowing you to access the data contained into SLIM. The second part is the 'DKS center': this is where you will generate configuration files from the data stored into SLIM. The third part is dedicated to the 'support' for your SLIS servers. And finaly, the fourth part is where the admin will manage SLIM-users and where users may change their preferences.

The main frame, is the place where tables, forms and outputs will appear. Users that are allowed to do some things may see some buttons that other users will not see. For example, if a user has not the right to edit ip addresses, he will not see the button 'EDIT'. By default, a user with no right at all, will be able to see the data (ecxepted passwords or keys) but not to edit, delete or create data.

4.2 The users

The slim administrator is a special user 'admin' that is allowed to do everything and to create/edit/delete other users. It's not recommended to work with this user. You should only use it to manage users. The slim administrator can create profiles and assign rights to profiles, and then give profiles to users. If a user has no profile or if his profile is empty, then, he will not be able to see or modify anything. In Slim 1.2, every right you

want to give to a user must be added to his profile. The 'group profiles' are like global profiles but the rights it they can containt are 'group rights' and are applied to a group of Slis. With this capabilities, you can create a limited group of slis that only a part of users will be able to see, modify...

Changing the password First of all, as you're logged as 'admin' for the first time, please, change your password: go into the last section called 'Prefs' and follow the link 'Password'.

Creating the first user Once logged as the user 'admin', expand the menu, and go at the bottom of it, into the 'users' section, click on 'accounts'. Click on 'New account' and fill in the form for the new user you are going to add: yourself. You have no global profiles for the moment, don't worry, you may edit this user later to add the profile.

- **Global rights** We manage permissions by the way of assigning a profile to a user. So, first create a global profile: link 'global profile', button 'new profile', enter a name and optionaly a comment, 'create', then click on its name when the list appears. Now, you have to add permissions to this profile: on the right, click on 'add'. This will list all the available global rights you can add. For example, if you want a user having this profile to be allowed to create a new ip address, then you add the 'create_ip_address' right. After having entered all the rights for this profile, you can edit your user(s) and select the profile.

- **Group rights** Now you can create group profiles, and assign rights to a user on a group of Slis. Create a group profile: link 'groups profile', button 'new profile', enter a name and optionaly a comment, 'create', then click on its name when the list appears. Now, you have to add permissions to this profile: on the right, click on 'add'. This will list all the available group rights you can add. For example, if you want a user having this profile to be allowed to view all Slis in a group, then you add the 'view_slis' right. After having entered all the rights for this profile, you can assign it to a user for a group of slis. This can be done by three ways. From the Slis group edit page, the account edit page or the groups profile edit page. You must select the 2 missing items to complete the triplet: account/group/profile.

4.3 Importing your initial data

You may start with no data into the database, and begin creating SLIS entries and related data for one given SLIS each time you need. But the goal of SLIM is to speed up the process of creating a lot of SLIS, so the normal use is to feed your database with the more known data as you can before starting creating SLIS entries.

There's a good chance that you already know what will be your sites, your networks, ip addresses, pops and backbones.

If you have official ip addresses, start by listing all the networks you have (for example 193.54.149.0/24, 195.221.234.0/24, etc). Then list all the private networks you are using or planning to use (for example 10.24.0.0/16 will be given to the SLIS servers in the NAT architecture, 172.20.20.0/24 will be used for tunnel ip addresses for the PPPOE SLIS servers, etc...) Then, you'll probably have to subnet those networks.

For example, if you subnetted 172.20.20.0/24 into 32 29 bits subnetworks, then, you'll have to create one network entry for each.

List all the ip addresses contained into those networks. You'll have to create one entry for each ip address. If you don't have a codification for your sites, imagine one codification! Feed the database with the more data as you can. If you have a doubt about the signification of a field, refer to the sample database provided into the `example.sql` file.

4.3.1 Feeding the database with SLIM importation functions

This is the 'easy but not efficient' way. It means that you can quickly import some data, but you'll may not be able to import everything by this way.

4.3.1.1 The import/export functions

The import/export functions appear in the menu when you expand it. There is one import link, and one export link for each main SQL table that is behind SLIM. But some data doesn't have import/export functions. If you want to import such data (like slim users or slis types for example), you'll have to do it by hand directly into your SQL server (see next section). Also note that the import functions doesn't manage the relations between tables. It's up to you to ensure that what you import into a table is coherent with the keys that points to other tables.

4.3.1.2 The input file

The input file for an import is a list of entries for the SQL table concerning the import. Each field is separated by a character that you may define in the form of the import function (separator). The first line contains the name of the field exactly as it is into the SQL model. For example, an import of the ip addresses looks like this:

Example 4.3.1

```
ip;ip_nat;nat_router_ip;comment;network_name;dns_id;mac
193.54.149.1;;;SERVEUR;193.54.149.0/24;1;
193.54.149.2;;;SERVEUR;193.54.149.0/24;2;
193.54.149.3;;;SERVEUR;193.54.149.0/24;3;
193.54.149.4;;;SERVEUR;193.54.149.0/24;4;
193.54.149.5;;;SERVEUR;193.54.149.0/24;5;
193.54.149.7;;;SERVEUR;193.54.149.0/24;7;
193.54.149.8;;;SERVEUR;193.54.149.0/24;8;
193.54.149.9;;;SERVEUR;193.54.149.0/24;9;
193.54.149.10;;;SERVEUR;193.54.149.0/24;10;
193.54.149.11;;;SERVEUR;193.54.149.0/24;11;
```

Tip: to quicly obtain the names of the fields, do an export, even if the table is empty. You'll obtain a file with the first line ready for your import.

4.3.1.3 Generating input files

Some data are repetitive, like all the ip addresses of a C class network. So, you can make scripts to generate import files. Here is a little perl script that will make an import for defining all the ip addresses of the 193.54.149.0/24 network:

```
#!/bin/perl

for ($i=0;$i<=255;$i++) {
print "193.54.149.$i;;;;193.54.149.0/24;\n";
}
```

4.3.1.4 Fixing sequences

SLIM tables often contains automaticaly incremented id numbers. Those are managed by postgresql sequences. In the model, you may find them with a value beginning with 'nextval::'. When you import new data, you perhaps generated new id numbers by yourself. So, we must tell to the database what will be the next value useable for each sequence. This is automaticaly done by a function that *you must call after each import*. This is done by expanding the menu and clicking on the first link at the top: 'FixKeys'. You must be logged as the administrator to do that.

4.3.2 Feeding the database with an SQL client

This is the more efficient way, but it requires that you're familiar to SQL syntax and that you refer to the SQL model detailed into the related chapter in this book.

Whatever the client you use, you'll have to log into the server before being able to send queries. The user/pass to use is the one you stored into `slim/include/config.php` when you installed SLIM.

4.3.2.1 The network SQL client

You may use what SQL client you want, but be care of security: if you connect via the network, you'll probably have to allow your client host to connect to your SLIM host via the postgresql port. So, you'll have to open the 5432/tcp port into your firewall config and to allow the client into the `pg_hba.conf` file. Just insert a line like this into your firewall script (`/usr/local/sbin/firewall.bash` if you installed SLIM by the automatic way):

```
iptables -A INPUT -p tcp -s <my_client_ip> --dport 5432 -j ↩
   ACCEPT
```

and the following line into `/var/lib/pgsql/data/pg_hba.conf`

```
host         all         <my_client_ip>     255.255.255.255 ↩
      password
```

Be carefull that this client/server connection is not secure as data will not be encrypted.

4.3.2.2 The command line SQL client

This is secure if you use the local client of your SLIM host throught an ssh session. This is efficient but maybe a bit difficult to use if your not a keyboard addict. To use the postgresql client on your SLIM host, just type in the command:

```
# psql -U slim slim
```

To import an SQL dump (a file containing a list of SQL queries), you can use the `\i FILENAME` macro of the psql command line client.

4.3.2.3 The PhpPgAdmin client (RECOMMENDED)

This is the recommended way as it is the better compromise: secure (if used over https), powerfull, simple to use, and very practical (as a simple browser becomes a powerfull SQL client).

So, we recommend you to download it from http://phppgadmin.sourceforge.net/- and install it in a subdirectory at the same level of the `slim` subdirectory under your htdocs tree. Don't forget to create an access protection for this directory into your apache configuration.

4.3.2.4 Generating the queries

Once you choosed your client, you can test it by creating a site, for example, by sending this query:

```
insert into site
(code,name,city,zipcode,phone,type,address,administrative\ ↩
    _status,district\_code,subtype)
VALUES ('test','Test Site','MyTown','38130','0404040','school ↩
    ','Av du
8 mai 1945','PU','','');}:>
```

If it works, you'll see this site into the SLIM interface. You are now ready to import your initial data with your SQL client.

Converted data You'll probably have to insert data that are comming from other databases. This is often the case for the list of sites (the site table) for example. The way I used was to obtain this list in an ascii format with a defined separator. Then, I created perl or awk scripts to convert this list in a continuation of SQL insert queries like the one above.

Example 4.3.2

For example, take the following input file:

```
0070052U|IEN|ANNONAY|188 AVENUE FERDINAND JANVIER||07100| ↩
    ANNONAY|0475337121|
0070053V|IEN|AUBENAS II|15 AVENUE DE SIERRE BAT 26||07200| ↩
    AUBENAS|0475356833|
0070054W|IEN|AUBENAS I|15 AVENUE DE SIERRE BAT 26||07200| ↩
    AUBENAS|0475350111|
0070055X|IEN|LE POUZIN|2 RUE DE HOTEL DE VILLE||07250|LE  ↩
    POUZIN|0475858681|
0070056Y|IEN|PRIVAS + AIS|PLACE ANDRE MALRAUX|627|07006| ↩
    PRIVAS CEDEX|0475669308|
```

We assume this file is `iens` into the local directory. The following awk command line will generate sql queries for inserting the data into the site table:

```
awk -F"|" '{print "insert into site(code,name,city) VALUES
    (\047" $1 "\047," $3 "\047,\047" $7"\047)"}' iens
```

Generated data Some data may be entirely generated. This is often the case for the ip addresses table.

Example 4.3.3

For example, the following perl script will generate the SQL queries for feeding the database with all the ip addresses of the network 193.54.149.0/24:

```
#!/bin/perl
for ($i=0;$i<=255;$i++) {
print "intsert into ipaddress (ip,network_name) VALUES  ↩
    ('193.54.149.$i','193.54.149.0/24');\n";
}
```

4.3.2.5 Fixing the sequences

Once you've inserted new data into tables, you have to fix the sequences that are used for incremented keys. See Fixing sequences from the previous section.

4.4 Setting up defaults and customs

SLIM allows you to set up default values for different kind of data:

- Table defaults: default values that will appear in some fields of a form when creating a new entry to speed up the capture
- File defaults: default values for variables of a generated file (DKS center) to overwrite the default variable of the matrix file

- DNS defaults: default DNS configuration matrix

SLIM also allows you to define some 'customizations':

- Views: predefined queries to speed up the browsing of some data
- Links: urls that may contain GET queries with a variable from SLIM, to create gateways to other applications
- Scripts: bash scripts that are called when generating config files
- Includes: some of the PHP includes are customizable

The defaults and customizations are editable by the SLIM administrator (user *admin*) only.

4.4.1 Table defaults

The tables defaults may all be found into the 'Defaults' section of the expanded menu

4.4.1.1 Slis

This table defines what the fields of a new SLIS form will contain by default depending on the type of the SLIS you choosed: when you create a new SLIS, you have to tell what the 'type' will be. Then, SLIM will search for all the entries in this table corresponding to this type. For each 'field' defined, it will fill it with the corresponding 'value' or 'method'. A method is a way to construct a value on the fly. There's currently 2 methods defined:

- val_today: the current date
- val_random_string: an 8 char random string. This is useful for passwords.

4.4.1.2 Tunel

This is the same thing, but for the tunel form that appears when you create a tunel from the SLIS edition.

4.4.1.3 Routes.tun

This is a bit different from the Slis or Tunel default because each entry of this table doesn't correspond to a field of another table, but to a new entry that will be created in the route table when a tunnel is created. The 'name' (first field) is a string you define here when you create a new route (you put what you want).

As routing is something often linked to the POP, the tunel routes default is keyed by the type AND the pop. So, the 2 fields of the middle consists of a double key.

The value (the last field) must be a network name in the same form in which the networks are defined into the network table.

See the interface reference guide for learning how the tunnel routes work.

4.4.2 File defaults

4.4.2.1 Setup.data

This works a bit like the Slis default, but instead of defining the defaults for fields of a table, you define defaults for variables of the `setup.data` file (this is the main config file of the setup disk for installing a SLIS).

This table is keyed on the Type and the Pop. It means that when you'll construct a new DKS (setup disk) from the 'DKS center', every variable defined here and corresponding to the type and pop (or to every type or pop if none is defined) of the selected SLIS will be replaced by the correspondig value in the setup.data of the obtained disk.

Note that this table has no effect to other table, unlike the defaults we talked about above. It only acts on the setup.data file when you make a DKS.

Some 'macros' can be used. They will be replaced by values from the database. Macros are:

- %slis_name: the name of the SLIS
- %pop_name: the name of the pop the SLIS depends on
- %site_name: the name of the site where the SLIS is installed

Note that if you want to put some special characters into the fields, (such as '$' into MD5 password chains), you must protect them with backslashes.

4.4.3 DNS defaults

I call this a 'matrix of default values'. The key is the type of the SLIS, but also the presence of a value in a cell.

This matrix will define the DNS associations to make for the four possible ip addresses of a SLIS, when the SLIS is created or edited. Each time you edit a SLIS, the matrix is computed and DNS for the concerned ip addresses is updated. When I say 'DNS', I talk about the DNS entries created into the database (tables dns_entry and ipaddress).

The ip addresses of a SLIS that may be DNS-configured are the 'Internet IP', the 'NAT IP', the 'PPP static IP' or the 'SLIS tunel IP'.

For a given 'type', the matrix is parsed until a line containg tokens for each ip used by the SLIS is found. For example, if the SLIS has a PPP static ip and a Tunel ip, but no Internet ip and no Nat ip defined, then the good line will be one containing nothing in the 2 first fields and something in the 3rd and 4th fields. This, for a given type or no type if it is not found.

The tokens are composed of macros. Those macros are:

- %slis_name: The name of the SLIS that is currently edited (without the domain)
- %domain: The value of the variable $def_domain that is defined into the configuration file (`include/config.php`).
- TO BE CONTINUED...

4.4.4 Views

4.4.5 Links

4.4.6 Scripts

custom_one.bash
custom_all.bash

4.4.7 Includes

slisexport.php

4.5 Creating your first SLIS into SLIM

4.6 Generating config files: the DKS center

4.7 Using maintenance tickets, deployment variables and SLIS statuses

4.8 Daily usage

Chapter 5

Interface reference guide

5.1 Admin

5.2 DNS

5.3 IP

5.4 Networks

5.5 ...

Chapter 6

Developpement